Article

Building Intelligent Natural Interfaces for Distributed SCADA Systems Using a Semantic Parsing Framework

Guicai Liang[1,*], Yurong Li [2] and Shijun Li [3]

[1] Information management center,Guangxi Technological College of Machinery and Electricity, Nanning 530007, Guangxi, China; lgc@gxcme.edu.cn
[2] Department of Mathematics and Computer Science, Shanxi Normal University Linfen College Linfen, 041000, Shanxi, China; zb$_l$*yrong*0000@126.*com*
[3] Computer School,Department of Artificial IntelligenceInstitute of Big Data Wuhan University, Wuhan, China; shjli@whu.edu.cn
* Correspondence: lgc@gxcme.edu.cn;

Abstract: Converting natural language into machine language that can be recognized by distributed systems is the core challenge of intelligent interactive interfaces and human-machine dialogue systems. The human-machine interface interaction of large distributed SCADA measurement and control system is tedious and the operation and maintenance cost is high, so it is significant to design an intelligent natural language interaction interface for distributed measurement and control system. In this paper, we design the intermediate language format of SCADA system, i.e., Key-value logic form, and then formulate the Text2SCADA framework and propose the TICS algorithm and SDPA algorithm, the former adopts the keyword extraction and cosine similarity optimization algorithm to complete the structured extraction of natural language for basic natural language instructions, and the latter adopts the keyword extraction and cosine similarity optimization algorithm to complete the structured extraction of natural language for relatively The latter one adopts the algorithm of dependent syntactic analysis for the structured representation of natural language instructions with relatively complex natural language instructions. Based on the above algorithms, a lightweight information extraction model based on DGCNN and probabilistic graph ideas is constructed, aiming to enhance the scientific generalization ability of the framework on unknown instruction sets. The experimental results show that the proposed intelligent natural language interface can better solve the human-machine interface interaction problem of distributed SCADA measurement and control system. The average accuracy, recall and F-value of instruction parsing reach 89.27%, 89.28% and 89.27%, respectively. The average response time is 1.593 s. Especially, it provides a more convenient means of interaction for industrial and agricultural information control.

Keywords: Natural language interface; Neural network language model; Dependent syntactic analysis; SCADA system; Human-computer interaction

1. Introduction

SCADA [1] is the most widely used data acquisition and monitoring control system in industry. Traditional SCADA system consists of dedicated monitoring computer, remote terminal unit (RTU), programmable logic controller (PLC), communication infrastructure, human-machine interface (HMI) [2].Link Inside Internet SCADA platform is based on 5G low latency, high bandwidth, high capacity, high reliability transmission technology, and local offload computing tasks of the MEC (multi-access edge computing) technology, for factory equipment interconnection, factory data collection, shop floor HMI to create an integrated support platform for smart factories. The function of the HMI (Human Machine Interface) is to display the received information in an easy-to-understand graphical way and to archive all the data received. However, once it involves more responsible control charts and cumbersome button controls, it will make the field staff make mistakes, and the way of human-machine interaction needs to be further improved.

Human-computer interaction [3] is an important application area of artificial intelligence, and with the development of artificial intelligence technology, more and more intelligent interaction products are appearing in people's daily life. In terms of interaction methods, "gesture interaction", "voice interaction", "AR interaction" and other emerging interaction methods are beginning to appear in the public eye, but due to the limitations of interaction efficiency and ergonomics, gesture interaction and other methods are less likely to become the most popular in the short term. However, due to the limitations of interaction efficiency and ergonomics, it is difficult for gesture interaction and other methods to become the mainstream human-computer interaction method in the short term, while products equipped with voice interaction capability have been widely concerned since their application on the ground. In terms of interaction means, users only need to dialogue with the relevant products to give commands, complete the task of playing music and controlling the home, which can truly liberate hands and improve the happiness index of life. However, in practical applications, it is often difficult to meet the complex requirements of users in various categories, which is rooted in the high complexity of natural language itself that makes the user's intention not well understood. Natural language understanding [4] (NLU ,Natural Language Understanding) task aims to provide computers with the ability to understand the user's language to make the next decision or complete the interactive action, which is an important task to be handled by products with voice interaction capability. Therefore, in this paper, based on the SCADA-oriented distributed measurement and control system, we hope to use natural language interaction technology to design a more intelligent and simple natural language human-machine interface for it, so as to improve the efficiency of human-machine interaction in operation and maintenance.

After decades of development since its introduction in 1970 [5], relational databases have become the mainstream tool for data storage and are widely used in government departments as well as in various fields such as business and academia. Structured query language (SQL) is an important means for users or applications to communicate with the database. The emergence of natural language interfaces, which break the interaction barrier between users and terminals, is one of the important branches in the field of artificial intelligence research, in which the implementation of database oriented interfaces has also received a lot of attention from researchers. Natural language interface to data-base (NLIDB) is an interface to query relational databases through natural language, which can eliminate the technical barriers encountered by users in querying databases, i.e., it removes the restriction that users must master the knowledge of SQL syntax, and realizes the function of querying databases through natural language descriptions. function. In a nutshell, the goal of NLIDB is to realize the conversion from natural language query to SQL.

Since the 1970s [6], there have been early NLIDB systems such as LUNARE [7], LADDER [8], Chat I 80 [9], ASK [10], etc.. Here, citing Affolter et al [11], these systems can be divided into four categories according to the technical approach: 1) keyword-based systems, represented by SODA [12], QUICK [13], etc.. These systems use the inverted index of basic data and metadata in the database as the retrieval object, and match them with natural language queries to identify the keywords mentioned in the query. However, the disadvantage of this method is that it cannot identify potential semantics that do not appear directly in natural language. 2) Pattern-based systems, represented by QuestIO [14], NLQ/A [15], etc., can map slightly complex natural language patterns to pre-specified query sentences. 3) Parsing-based systems, represented by the following methods Parsing-based systems, represented by ATHENA [1 6], NaLIR [17], etc., introduce more natural language processing techniques, such as parsing natural language interrogatives with the help of grammar analysis trees. Therefore, such methods can further reflect the semantics in the question into a predefined SQL template.4) Grammar-based systems, represented by Ginseng [18], TR Discover [19], etc., which have a set of predefined grammar rules that restrict the user's input behavior to form a natural language query in a standardized format, which is easy to analyze systematically. These four types of methods can be summarized as rule-based generation methods. Systems based on these four types of methods can only achieve sQL output according to a fixed database format and a predefined query template. However, the prevalence of multiple ambiguities and description diversity in natural languages leads to the fact that the pre-defined matching rules or query templates cannot cover more and more complex natural language queries in practical applications. Most of the above methods can only deal with fixed domain databases, i.e., NLIDB can only be designed and implemented according to a fixed table structure, so the system generally cannot be ported to other databases. The main reason why there is no general commercial solution or prototype system yet is that there is no NLIDBE for different database table structures [20]. The application of deep learning in recent years has greatly improved the technology in many artificial intelligence fields such as speech recognition, visual object recognition, and bioinformatics [21]. Deep learning models are based on neural network implementations, and one important class of network structures, recurrent neural networks, plays an important role in processing sequential data such as text and speech. The RNN-based encoder-decoder framework is an end-to-end deep neural network structure that is widely used in many natural language processing tasks such as machine translation, semantic analysis, text summarization, and dialogue generation. The goal of the semantic parsing task is to transform natural language into a 109ical form. This logical form is a semantic representation that is driven by natural language and can map the meaning of natural language [22]. For example, programming languages such as Lambda Calculus, SQL and mechanical control instructions, or general-purpose programming languages such as Python and Java, all belong to the category of logical forms. Therefore NLIDB implemented based on the encoder-decoder framework can be considered as a class of semantic analysis models where the loser is a natural language query that is analyzed by an encoder-decoder.

Therefore this paper contributes as follows :

- An intermediate language Key-value form is designed to be used as an intermediate language logical form for controlling the SCADA system.
- The Text2SCADA natural language framework is proposed for converting natural language into a Key-value intermediate language logical form that can be understood by the SCADA system.
- A lightweight information extraction model based on DGCNN and probabilistic graphs is constructed to take the original natural language as input and the Key-value intermediate language logical form as output to build an end-to-end neural network model for recognizing more unknown natural language commands and enhancing the scientific generalization capability of the overall framework.

2. SCADA system intermediate language format

The HTTP address is divided into three parts in a SCADA system: view absolute path , view channel and IDserver IP address. The first half of them, such as host-id, PATH and Plugs, are irrelevant variables that we do not need to care about. cnlNum corresponds to the channel number (i.e., variable name) of the data collection, and viewID means the number of the view (i.e., location name), aday, month, and year represent specific dates. These three attributes allow retrieving the view number and the variable value under the channel number for a certain date. Figure 1 presents a command parameter to automatically fill the sketch. For example, in a smart home system, the three important information of action, place and object need to be extracted. When we input the natural language command "turn on the air conditioner in the living room", it is necessary to convert it into a structure of data as " Object = lamp, Location = living room, Action = turn on". Then, SCADA reads the value of Object, Location and other attributes in the above structure to retrieve the target and perform the corresponding action.

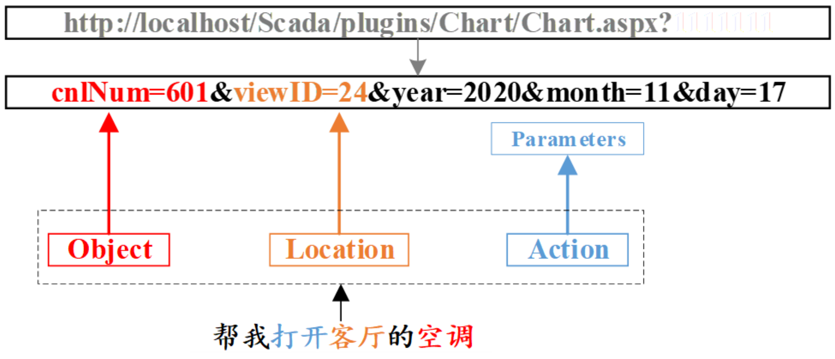
http://localhost/Scada/plugins/Chart/Chart.aspx?
cnlNum=601&viewID=24&year=2020&month=11&day=17
Parameters
Object
Location
Action
帮我打开客厅的空调

Figure 1. HTTP adress for living room thermo data

3. Text-to-SCADA architecture

To implement natural language manipulation of SCADA systems, the key is to extract object indicators, location indicators, and action indicators from natural language commands. In this paper, we design a Text-to-SCADA architecture as shown in Figure 2. The natural language is converted into a machine language that can be recognized by the SCADA system. Firstly, the commands are identified by basic and complex commands, and if they are basic natural language commands, Then it is a good choice to use TICS (TF-IDF+Cos_Sim) algorithm for semantic parsing, otherwise SDPA (Semantic dependency parsing algorithm) is recommended for parsing.

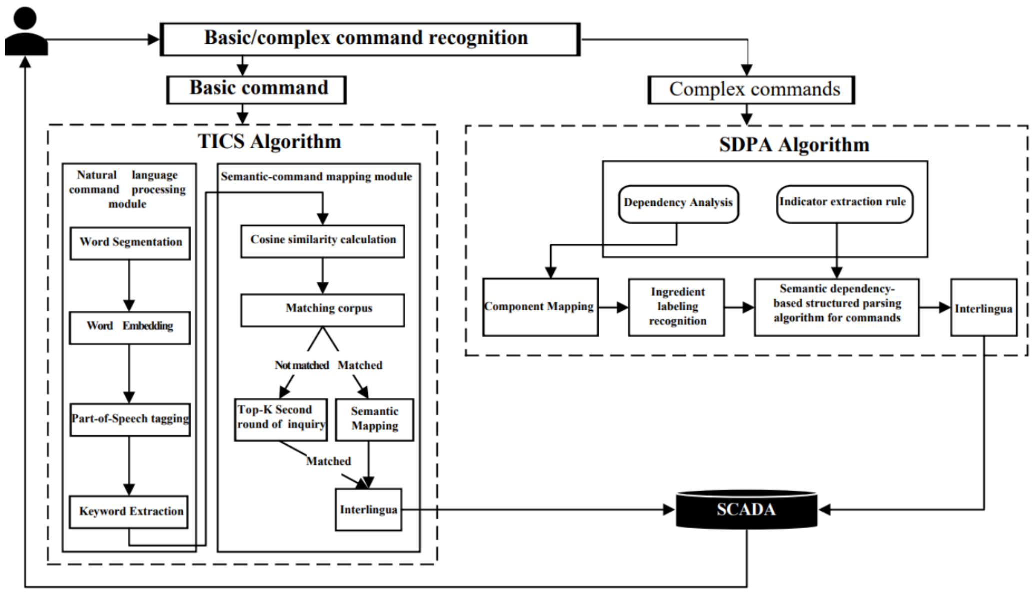
Basic/complex command recognition
Basic command
Complex commands
TICS Algorithm
Natural language command processing module
Word Segmentation
Word Embedding
Part-of-Speech tagging
Keyword Extraction
Semantic-command mapping module
Cosine similarity calculation
Matching corpus
Not matched
Matched
Top-K Second round of inquiry
Semantic Mapping
Matched
Interlingua
SDPA Algorithm
Dependency Analysis
Indicator extraction rule
Component Mapping
Ingredient labeling recognition
Semantic dependency-based structured parsing algorithm for commands
Interlingua
SCADA

Figure 2. Text-to-SCADA Framework

3.1. *Basic and complex instruction recognition*

Complex natural language commands can be seen as consisting of basic natural language commands. The natural language instructions are lexically annotated using the dependency analysis system DDParser.The results of the word labelling show that the items tagged with /c are basically conjunctions within the command. Words, phrases and sentences connected by parallel conjunctions,such as "then", "and", "or", "so " and so on.Nowadays, the common approach to determine the presence of sub-instructions is to consider whether the input natural language instruction contains concordant conjunctions or not. If a coordinating conjunction such as "and", "or", "then" is found, then the instruction is considered a complex natural language instruction. For example, if the input natural language instruction is "turn on the lamp in the living room and turn off the hood in the kitchen, then check the temperature in the bedroom", the labeled instruction is shown in Table 1.

Table1. Example of command lexical annotation

子指令 (Sub-command)	标注情况 (Labeling status)
打开客厅的台灯和 (Turn on living room of table lamp and)	打开/v 客厅/n 的/u 台灯/n 和/c (Turn on /v living room /n of /u table lamp/n and /c)
关闭厨房的油烟机 (Turn off the kitchen hood)	关闭/v 厨房/n 的/u 油烟机/nz (Turn off/v the/n kitchen/u hood/nz)
然后查看卧室的温度 (Then check the temperature of the bedroom)	然后/c 查看/v 卧室/n 的/u 温度/n (Then/c check/v the temperature/n of/u the bedroom/n)

3.2. *TICS algorithm*

The natural language instructions go through four steps: word division and word embedding, stop word filtering, lexical annotation and keyword extraction. The process is shown in Table 2.

Table2. Natural language instruction processing flow

输入 (Input)	太热了，帮我打开客厅的空调 (It's too hot, help me turn on the air conditioning in the living room)
Step.1 分词 (word division and word embedding)	太热/了/帮/我/打开/客厅/的/空调 (It's too hot/,/ help me /turn on/the living room air conditioning/ air conditioning/)
Step.2 停止词过滤 (stop word filtering)	太热/打开/客厅/空调 (too hot/turn on/the living room/air conditioning)
Step.3 词性标注 (lexical annotation)	太热, a/打开, v/客厅, np/空调, ns (too hot, a/turn on, v/the living room, np/air conditioning, ns)
Step.4 关键词提取 (keyword extraction)	打开, v/客厅, np/空调, ns (turn on, v/the living room, np/air conditioning,ns)

3.2.1. Natural Language Processing

Firstly, the distributed representation of words is completed using the Skip-gram model in Word2vec to implement the word embedding algorithm, which has to be processed by Chinese word separation and relies on the Chinese word separation tool Jieba to implement. After the word separation, the deactivation word list of HIT is loaded to achieve filtering of low-value words and phrases in natural language instructions. In order to limit the scope of word performance and improve the extraction of keywords, lexical annotation of instructions is required. The lexical annotation in this paper is benchmarked against the PKU lexical annotation set. Firstly, a dictionary containing 98 words is defined to match lexical properties. At the same time, lexical annotation is done in the separation process. After the completion of lexical annotation, keyword extraction is performed.Stop word elimination, word frequency-inverse text frequency algorithm and lexical similarity calculation are used to achieve the extraction of relevant attributes describing the controlled object in the instruction. Term Frequency (TF), which indicates the frequency of keyword w occurring in the file, is computed as :

$$TF_{W_iD_i} = \frac{count(w)}{|D_i|}$$

The number of occurrences of keyword w is represented by count(w), and the number of all words in the file is represented by D_i.

3.2.2. Semantic-command mapping

（1）Calculation of cosine lexical similarity

In this paper, we train the word vectors of high-frequency words in industrial and agricultural fields with the help of neural network language model in Word2vec, and also cluster the words by the cosine similarity between the word vectors to eliminate the phenomenon of multiple words with one meaning.Word vector sums using the cosine similarity calculation :

$$cos < x_i, y_i >= \frac{\sum_{i=1}^{n} x_i \times y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \times \sum_{i=1}^{n} y_i^2}}$$

[0,1] is the value interval of cosine similarity. （2）Instruction matching and mapping

After processing the words in the natural language command sequences extracted by keywords to obtain their vector forms, the cosine similarity with the words in the corpus is calculated and a threshold p is set, and when the threshold is exceeded, the two words are judged to be close. After matching the command keywords in the corpus, it needs to be transformed into the intermediate language data structure defined in the section "Intermediate Language Format for SCADA Systems" for the SCADA system to read. As long as the query can be customized by the user as an index table (e.g. Table 3), the keyword attributes (location, object, action) of the measurement and control objects are extracted from the keyword extraction module as keys to query the corresponding values and fill in the corresponding http address or call JavaScript methods to achieve the query or control operation.

Table3. Parameter index table

Keywords	ViewID
卧室(Bedroom)	1
客厅(Living room)	2
厨房(Kitchen)	3
卫生间(Restrooms)	4
......	......

In summary, Algorithm 1 describes the semantic parsing process of the TICS algorithm.

Algorithm 1 TICS Basic Natural Language Instruction Parsing Algorithm

```
Input: String, a sequence of basic natural language instructions.
Output:Structured set of key-value pairs [{Object:word_1},{Location:word_2},{Action:word_3}]
1. RESULT = { Ø }/*initialize an empty set*/
2. StringList = Keyword_Extract(String,'TF') /*TF-IDF keyword extraction*/
3. FOR List IN StringList/*Iterate through the extracted keyword list*/
4.     IF List[0] IN Location_dict OR List[1] == 'ns' /*If the keyword is in the location attribute dictionary or
       the lexical property is ns, then the word is identified as a location attribute*/
5.         location ← List[0]
6.     ELSE IF cos_sim>=0.65
7.         Location ← List[0]
8.     END IF
9.     IF List[0] IN Object_dict OR List[1]=='nz'OR'nc'
10.          Object ← List[0] 11.
11.    ELSE IF cos_sim>=0.65
12.        Object ← List[0]
13.    END IF
14.    IF List[0] IN Action_dict OR List[1]=='v'
15.        Action ← List[0] 16.
16.      ELSE IF cos_sim>=0.65
17.        Action ← List[0]
18.    END IF
19. RESULT←[{Object:word_1},{Location:word_2},{Action:word_3}]/*Update result*/
20. END FOR
```

3.3. *SDPA algorithm*

The SDPA algorithm targets complex natural language instructions and uses DDParser to perform dependency syntactic analysis of complex natural language instructions. The natural language instructions are identified by component mapping and component labeling, and the dependencies between word prototypes in complex instructions are analyzed and indicator extraction rules are defined to achieve semantic parsing of complex natural language instructions. In this paper, a bottom-up parsing framework is constructed, as shown in Fig 3.

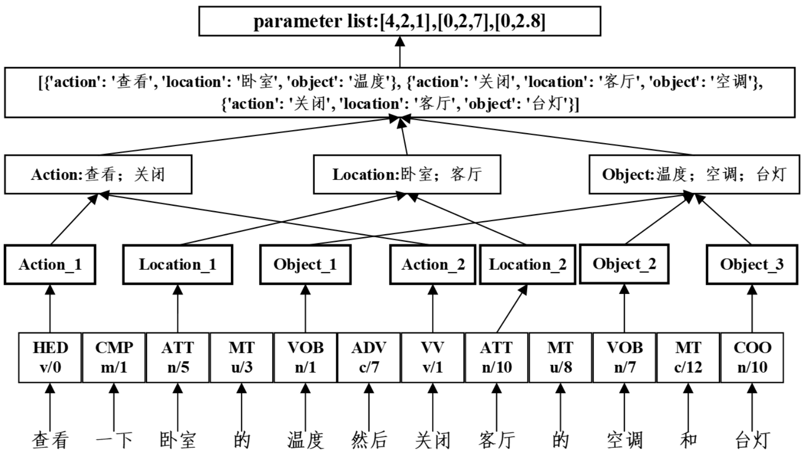

Figure 3. SDPA parsing framework

3.3.1. Dependent syntactic analysis

The complex natural language command "Turn on the air conditioning in the shed and then turn off the lights in the shed" generates the dependencies as listed ['word': ['先'('first'), '打开'('open'), '大棚'('shed'), '的'('of'), '空调'('air conditioner'), '然后'('then'), '关闭'('off'), '田间'('field'), '的'('of'), '灯光'('light')], 'postag': ['d', 'v', 'n', 'u', 'n', 'c', 'v', 'n', 'u', 'n'], 'head': [2, 0, 5, 3, 2, 7, 2, 10, 8, 7], 'deprel': ['ADV', 'HED', 'ATT', 'MT', 'VOB', 'ADV', 'VV', 'ATT', 'MT', 'VOB']]。The natural language instructions were component mapped into four metrics, namely "word", "postag", "head" and "deprel". These are, in turn, the result of the natural language instruction, the lexicality of the word, the positional dependencies between the semantics of the words, and the semantic relationships. After summarising complex natural language instructions, four main types of semantic dependencies are found to be commonly found in instructions: corelations (HED), definite middle relations (ATT), verb-object relations (VOB) and concurrent relations (COO). Examples of common dependencies in natural language are given in Table 4.

In natural language instructions, the verb is usually the core of the sentence and does not depend on any word, so the number under "打开"("Open") is 0, i.e. it does not form a dependency with any word and is defined as a ROOT node. In this case, the number under "客厅"("Living Room") is 5, i.e. it forms an ATT with the fifth word "空调"("Air Conditioning"). The number under "客厅"("Living Room") is the definite article of "空调"("Air Conditioning"), and the number under "卧室"("Bedroom") is 10, which is the definite relationship (ATT) with the tenth word "灯光"("Lights"). The number 10 under "bedroom" is the same as the 10th word "灯光"("Lights"). Based on this property, it is possible to extract the prototype of the ATT as a location property in the intermediate language data structure paradigm. In the example, the number under "空调"("Air Conditioning") is 2, i.e. it forms a verb-object relationship (VOB) with the second word "打开"("Open"), and the number under "灯光"("Lights") is 7, i.e. The number under "灯光"("Lights") is 7, which is the verb-object relationship (VOB) with the seventh word "关闭"("Close"). Based on this property, the word prototype corresponding to the VOB can be selected as an object property in the intermediate language data structure. In the command "打开厨房的油烟机和洗碗机"("Turn on the kitchen hood and dishwasher."), the words "油烟机"("Hoods") and "洗碗机"("Dishwasher") form a parallelism (COO), i.e. a relationship between words of the same type, so In this case, the prototypes of the COO can be extracted and merged into the Object property. For the Action attribute, we can consider the VOB relationship. After finding the object attribute, we can extract the keywords on which the object depends as the action attribute according to the dependency between the verb and the object, e.g. "关闭灯光"("Turn off the lights"), "关闭"("Close") and "灯光"("Lights") form a verb-object relationship, so the action attribute can be identified as "关闭"("Close"). Thus, after the above steps the syntax is combined to obtain a paradigm for the intermediate language data structure of the complex natural language 'first turn on the air conditioner in the living room and then turn off the lamp in the bedroom': i.e. ['action':'打开'('Open'), 'location': '客厅'('Living Room'), 'object': '空调'('Air Conditioning'), 'action': '关闭'('Close'), 'location': '卧室'('Bedroom'), 'object': '灯光'('Light')]. The final semantic-command mapping completes the parameter transformation. The dependency tree for the commands 'turn on the air conditioner in the living room and then turn off the lights in the bedroom' and 'turn on the kitchen hood and dishwasher' is shown in Figure 4.

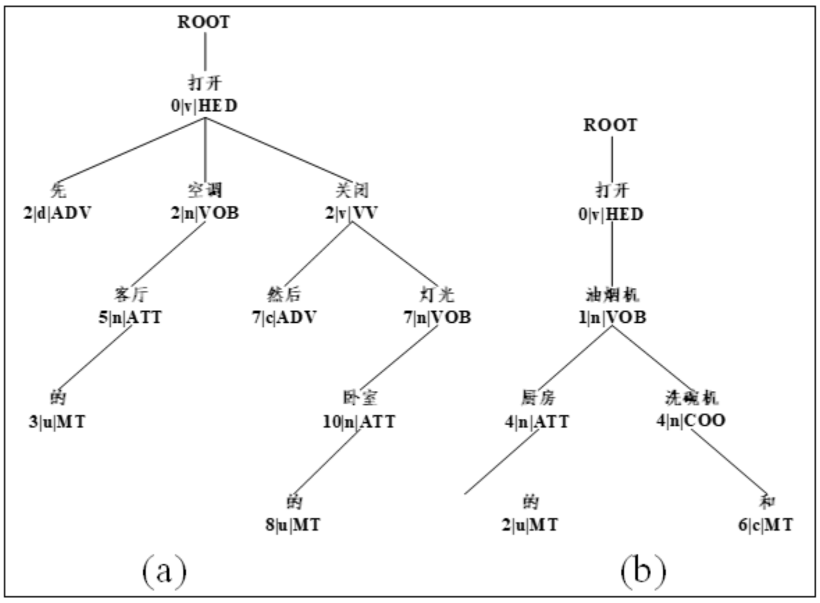

Figure 4. Dependency syntax diagrams

Table4. Example of dependency relationship

关系	描述	举例
HED	指整个句子的核心	打开卧室的台灯（ROOT，打开，HED）
ATT	定语与中心词之间的关系	关闭大棚的传感器（传感器，大棚，ATT）
VOB	宾语与谓语之间的关系	请帮我打开厨房的油烟机（打开，油烟机，VOB）
COO	同类型词语之间的关系	打开卧室的空调和台灯（空调，台灯，COO）

3.3.2. Key attribute extraction rules

Key attributes in complex natural language instructions are extracted through dependent syntactic analysis as well as lexical features and populated into the intermediate language data structures defined in Section 'action': ", ' location': ", 'object': ". The extraction steps are as follows:
(1) Generating dependency analysis results for complex natural language instructions.
(2) Traversing the results of the dependency analysis and extracting key indicators based on semantic relations as well as lexical features.
(3) automatically populating them with pre-defined intermediate language data structures.

When utilizing semantic features for key attribute extracting, three types of semantic relations need to be searched for: definite in relation (ATT), verb-object relation (VOB),juxtaposition relation (COO).The following three rules are required to be obeyed for key indicator extraction :

The first rule: when the command contains multiple verbs, the verbs will have corelations (HED) and conjunctive-predicative relations (VV), which are very unfavorable for direct extraction. At this time, we should first search for the verb-object relationship (VOB), then according to the language habits and lexical features, find the corresponding verb with the help of the object.

The second rule: The definite article is mainly a modifier, and this paper considers the semantic parsing responsible for manipulating instructions, of which the position information is usually represented by the definite-in relationship (ATT). Therefore it can be extracted as a positional attribute.

The third rule: the juxtaposition relation (COO) has the same role in the manipulation instruction, i.e., the manipulation object has the same action, and for this relation, so that its manipulation action and position information are consistent, and the word prototypes corresponding to the verb-object relation (VOB) can be combined.

Algorithm 2 describes the semantic parsing process of the SDPA algorithm.

Algorithm 2 SDPA complex natural language instruction parsing algorithm

Input: ParserDict, a dictionary of complex natural language instruction dependencies, containing {word, postag, head, deprel} . where word denotes the word prototype, postag denotes the lexical property, head denotes the positional dependency between two words, and deprel denotes the dependency between two words.

Output: a structured set of key-value pairs [{*Object:word_1*},{*Location:word_2*},{*Action:word_3*}]

```
1.   RESULT={ Ø }/* initializes an empty set */
2.   number = -1/* initialize count variable */
3.   WordList = ParserDict['word']
4.   PostagList = ParserDict['postag']
5.   HeadList = ParserDict['head']
6.   RelationList = ParserDict['deprel']
7.   FOR relation IN RelationList
8.           number = number+1
9.           IF relation == 'ATT' /* iterate through the fixed relations */
10.             Location ← WordList[number]
11.             Object ←WordList[HeadList[number]-1]
12.          ELSE IF
        (relation=='VOB') OR(relation=='COO')
13.               IF relation =='VOB'
14.                    Action←WordList[HeadList[number]-1]
15.                  ELSE    Object←WordList[number]
16.                END IF
17.             END IF
18.  RESULT←[{Object:word_1},{Location:word_2},{Action:word_3}]/* update result */
19.  END FOR
```

3.3.3. Parsing algorithm optimization strategy

By analysing the noisy data that often occurs in complex natural language for dependency syntax analysis, failed extractions are automatically eliminated. The Chinese language itself is very complex and the dependencies formed by the same word in different contexts may be different, e.g. "请帮我打开卧室的台灯"("Please turn on the bedroom lamp for me.") and "打开卧室的台灯"("Turn on the desk lamp in the bedroom.") are two natural language commands that convey exactly the same meaning, but The two natural language commands have the same meaning, but the dependency of the keyword '打开'('Open') is different. In the first command, '打开'('Open') and '帮'('Help') form a double object relationship (DOB). In the second instruction '打开'('Open') and 'ROOT' form a core relationship (HED). Similarly, in the same natural language command, the word prototypes for the same dependency may not be the same, e.g. "帮"('Help') and "台灯"("Desk Lamp") in "请帮我分别打开卧室的台灯和客厅的空调"("Please help me turn on the desk lamp in the bedroom and the air conditioner in the living room respectively."). Both "帮"("Help") and "台灯"("Desk Lamp") in "请帮我分别打开卧室的台灯和客厅的空调"("Please help me turn on the desk lamp in the bedroom and the air conditioner in the living room respectively.") form a verb-object relationship (VOB) with other words. In addition, human speech is mixed with other words, e.g. "打开卧室的扫地机器人, 卫生需要保持干净"("Turn on the bedroom sweeper, hygiene needs to be kept clean"), the result of Algorithm 2 is: ['action': '打开'('Open'), 'location': '卧室'('Bedroom'), 'object': '扫地机器人'('Floor Sweeper'), 'action': '打开'('Open'), 'location': '卧室'('Bedroom'), 'object': '保持'('Keep')]. It can be seen that the second dictionary is redundant and mistakes '保持'('Keep') for an object attribute. The main reason for this is that "保持"("Keep") and "需要"("Need") form a verb-object relationship (VOB), and "扫地机器人"("Floor Sweeper") and "打开"("Open") also form a verb-object relationship (VOB). is also in a VOB relationship with "打开"("Open"). Algorithm 2 incorrectly identifies "保持"("Keep") as an object attribute and completes the indicator extraction. Therefore, in the optimization algorithm module, the redundant and invalid information in the structured results needs to be automatically removed, and the incorrectly extracted information needs to be eliminated by combining lexical features. This improves the accuracy of instruction parsing and enhances the scientific, applicability and extensibility of the SDPA algorithm for structuring and parsing complex natural language instructions.

Algorithm 3 Complex instruction parsing result optimization algorithm

Input: RESULT

Output: Optimized RESULT

```
1.    i = 0
2.    Object_value = { Ø }/* initialize an empty list */
3.    IF Action != Ø   AND   Location != Ø AND Object != Ø
4.    FOR item IN RESULT
5.            Object_value←item(key_value)/* store the value of the indicator representing the object
property in the list */
6.    END   FOR
7.    FOR   string1   IN   WordList
8.        i = i+1
9.    FOR string2 IN Object_value
10.        IF string2 == string1
11.            IF (PostagList[i-1]!='nz') AND (PostagList[i-1] !='n')
12.                Delete(dict←object)/* object attributes if not nz or n, then the object who is in the
dictionary to delete */
13.            END IF
14.        END IF
15.     END FOR
16.   RESULT ← RESULT
17.   RETURN RESULT
18.   END FOR
```

3.4. Lightweight information extraction model based on DGCNN and probabilistic graphs

Information extraction (IE) is a text processing technique that extracts factual information such as entities, attributes, relationships and events from natural language text.Therefore, in this paper, after completing the above discrete natural language to intermediate language transformation, a deep learning approach is adopted to train the generated natural language instruction pairs, and the data structure of the training is shown below.

{ "natural language command": "请去把客厅的吊灯打开"("Please go turn on the chandelier in the living room."),

"Intermediate Languages": [

["object ", "吊灯"("Chandelier")],

["location", "客厅"("Living Room")],

["action", "打开"("Open")]

]

}

Overall it is the input of a natural language instruction, and then the model is trained to get the output in the form of an intermediate structured language, i.e., a triplet. Where the triad is of the form (0, l, a), so this paper uses the idea of probabilistic graphs to design an efficient extraction scheme using CNN-Attention as the core architecture of the model. In the seq2seq scheme, the decoder is mainly performing probabilistic modeling, as shown in the equation.

$$P(y_1, y_2 ..., y_n|x) = P(y_1|x)P(y_2|x, y_1)...P(y_n|x, y_1, y_2, ..., y_{n-1})$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 February 2023 doi:10.20944/preprints202302.0325.v1

The actual prediction is done by first predicting the first word by x, then assuming that the first word is known to predict the second word, and so on, until the end marker appears. Therefore, when predicting triads, this paper considers this idea by defining the following equation.

$$P(o, l, a) = P(o)p(a|o)P(l|o, a)$$

In other words, we can first predict o, then pass in o to predict the a corresponding to that o, and then pass in o, a to predict the relationship l of the passed in o, a. In practice, we can also combine the prediction of l, a into one step, so the total step only requires two steps: first predict o, then pass in o to predict the a and l corresponding to that o. Thus the model architecture is shown in Figure 5. 338 339 340 341 342

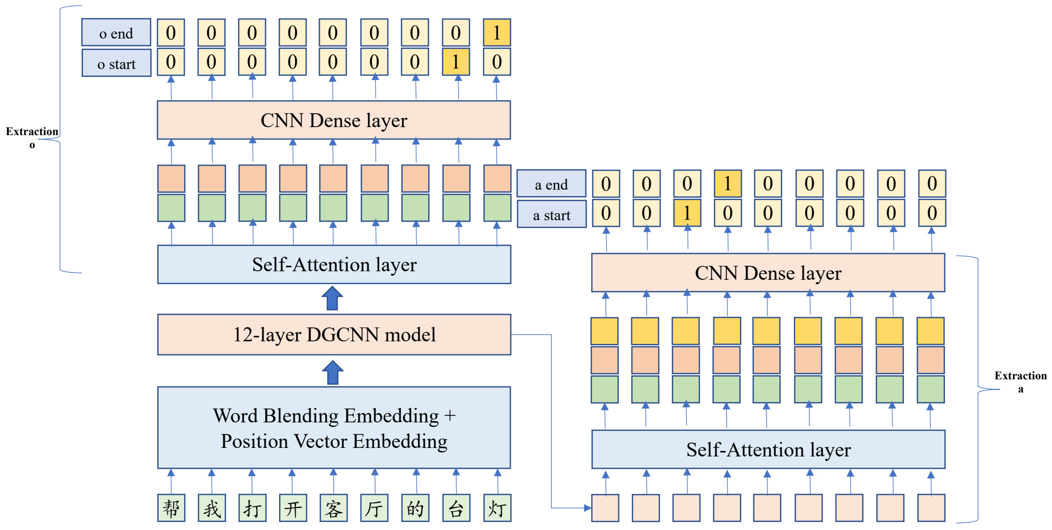

Figure 5. Architecture diagram of lightweight information extraction model based on DGCNN and probabilistic graph

343 The processing flow of the model is shown below.

344 1. input word id sequence, and then get the corresponding word vector sequence by Word Mixing Embedding, and then add Position Embedding. 345 346

347 2. input the "word-word-position Embedding" into the 12-layer DGCNN for encoding, and get the encoded sequence (note as H). The details of DGCNN are shown in the Figure 6. 348 349

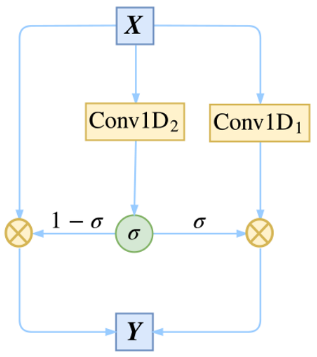

Figure 6. DGCNN

The equation is shown below.

$$Y = X \otimes (1 - \sigma) + \text{Conv 1D}_1(X) \otimes \sigma$$
$$\sigma = \sigma(\text{Conv 1D}_2(X))$$

3.after passing H into a layer of Self-Attention, the output is spliced with a priori features (a priori features can be added or not, the construction method will be described in detail later).

4. the spliced results will be passed into the CNN, Dense, using the "half pointer - half labeling" structure to predict the first and last position of o.

5.training randomly sampled a labeled o, and then H corresponding to the subsequence of this o passed into a two-way LSTM, to get the o encoding vector, and then add the relative position of Position Embedding, to get a vector sequence of equal length with the input sequence.

6.after passing H into another layer of Self-Attention, the output result is spliced with the vector sequence and prior features output in step 5.

7.Pass the spliced result into CNN and Dense, and for each kind of l, construct a "half-pointer-half-labeling" structure to predict the first and last positions of the corresponding a. In this way, both a and l are predicted at the same time.

4. Experimental setup and analysis of results

4.1. Test dataset construction

Text-to-SCADA is a domain-specific framework for natural language instruction parsing, and for the application domain proposed in this paper, we constructed the Chinese test instruction dataset - CNLQTS. regarding the corpus collection instead of using direct integration of conversational text, various instructions that may appear in the SCADA system were predefined and then The research staff crowdsourced the data and unfolded it into natural, spoken sentences. This construction method is effective in improving Text2SCADA's ability to understand natural language instructions in a given domain. First, we define a command that conforms to the intermediate language specification mentioned in the SCADA system's intermediate language format 'object': '空调'('Air Conditioning'), 'location ': '客厅'('Living Room'), 'action': '打开'('Open'). The pseudo-linguistic problem description was then rewritten as a natural language problem E = '打开客厅空调'('Turn on the living room air conditioner'), and the utterance E was expanded into a more complex, colloquial natural language utterance. A case study from the corpus is shown in Table 5.

Table5. Corpus examples

中间语言 (Intermediate Language)	自然语言 (Natural Language)
object = 吊灯, (object = chandelier)	语句①帮我打开客厅吊灯 (Statement ①Help me turn on the chandelier in the living room)
location = 客厅, (location = living room)	语句②把客厅的吊灯打开 (Statement ②Turn on the chandelier in the living room)
action = 打开 (action = open)	语句③请去把客厅的吊灯点亮 (Statement ③Please go and light up the chandelier in the living room)

The final CNLQTS dataset provides a collection of 600 natural language commands from different domains, such as agriculture (200), industry (200) and smart home (200). Each domain contains 100 basic natural language instructions and 100 complex natural language instructions. Experimental tests were conducted using various types of data as shown in Table 6.

Table6. Example of CNLQTS dataset

领域 (Fields)	指令示例 (Command example)
农业 (Agriculture)	大棚的温度是多少? (What is the temperature of the greenhouse?)
农业 (Agriculture)	请帮我查看田间的甲醛含量是多少? (Please help me to check what is the formaldehyde level in the field?)
农业 (Agriculture)	打开大田的灯光，然后查看一下温室的温度。 (Turn on the lights of the big field and then check the temperature of the greenhouse.)
农业 (Agriculture)	请帮我分别打开大棚的灯光和空调。 (Please help me to turn on the lights and air conditioning of the shed separately.)
工业 (Industrial)	机房现在甲醛浓度是多少? (What is the current concentration of formaldehyde in the server room?)
工业 (IndustrialIndustrial)	中控机房的二氧化碳浓度是多少? (What is the CO2 concentration in the central control room?)
工业 (Industrial)	先打开机房的吸尘机再关闭无尘室的空调。 (Turn on the vacuum machine in the machine room and then turn off the air conditioning in the clean room.)
工业 (Industrial)	请帮我查看一下中控机房的甲醛浓度。 (Please help me to check the formaldehyde concentration in the central control room.)
智能家居 (Smart Home)	打开厨房的油烟机。 (Turn on the kitchen hood.)
智能家居 (Smart Home)	查看卫生间的温度是多少? (Check what the temperature of the bathroom is?)
智能家居 (Smart Home)	先打开卧室的台灯，再关闭卫生间的浴霸。 (Turn on the desk lamp in the bedroom first, then turn off the bathroom bath.)
智能家居 (Smart Home)	打开客厅的扫地机器人和吊灯。 (Turn on the floor sweeper and chandelier in the living room.)

4.2. Text-to-SCADA framework evaluation indicators 389

The evaluation method of Text2SCADA performance mainly consists of three methods: Precision, Recall and F-score values, which are obtained by manual determination. A total of 600 natural language instructions were constructed for the dataset, and the number of instructions was considered to influence the experimental results, and different sets of instructions were selected to test the performance of the algorithm according to the specific experiment. The definitions of the relevant variables are given next:

A denotes the indicators correctly identified and the number of corresponding indicator values.

B denotes the total number of records with structured results.

C denotes the total number of indicators contained in the original natural language instructions.

The correlation is then calculated as follows:

$$P = \frac{A}{B}$$

$$P = \frac{A}{C}$$

$$P = \frac{2}{P + R}$$

4.3. Training and testing

4.3.1. Word embedding

The skip-gram model was used, the loss function was cross-entropy, the word vector dimension was set to 150 and the window size was 8. The stochastic gradient descent method was used for training, and the loss curve during training is shown in Figure 7.

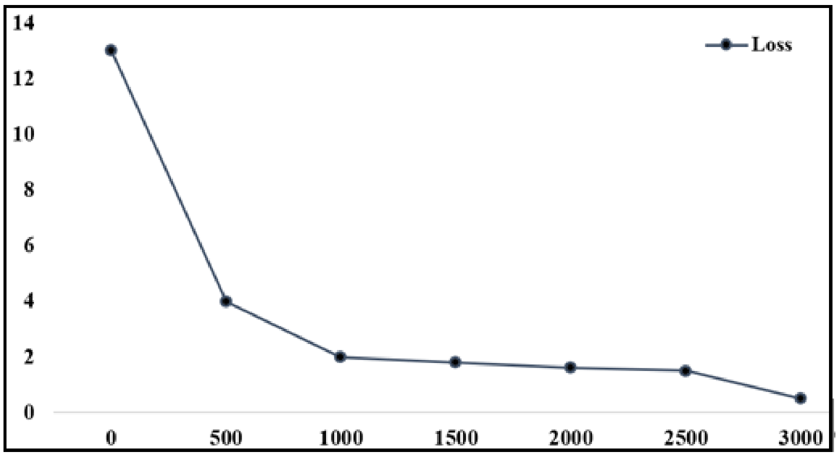

Figure 7. Loss curve

Once the vector form of Chinese words is obtained, the similarity between the input words and the individual example sentences in the corpus can be calculated by using the formula, and the words with the highest similarity and the same lexical nature are taken to match the words in the preposterous corpus. The semantic-command mapping module then retrieves the attributes such as object and location described by the command and maps it to an http address or calls a JavaScript method to execute the command.

4.3.2. Experiments to test basic natural language instruction parsing algorithms

To evaluate the performance of the TICS basic natural language instruction parsing algorithm, experiments were conducted using basic natural language instructions aggregated from the CNLQTS dataset. Using control variables as well as random assignment, 50, 100, 150...300 instructions were sequentially entered for testing. The values of the evaluation metrics obtained and the maximum time to return results are shown in Table 7. Where Figure 8. clearly reflects the variation of each assessment indicator.

Table7. Basic natural language instruction test results

No.	Number	Precision	Recall	F-score	Time
1	50	92.15%	94.25%	93.19%	0.82s
2	100	91.57%	92.14%	92.85%	0.74s
3	150	91.24%	91.71%	91.47%	0.67s
4	200	90.25%	91.14%	90.19%	0.89s
5	250	89.65%	88.54%	89.06%	0.87s
6	300	87.58%	87.91%	87.73%	0.73s
Avg		**90.37%**	**90.79%**	**90.58%**	**0.79S**

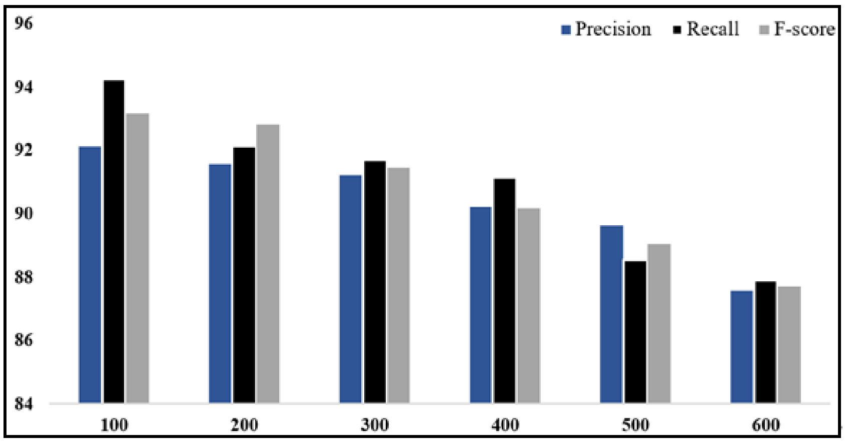

Figure 8. TICS performance evaluation bar chart

As you can see in the figure :

- From the performance on the test set, the average values of the three main assessment indicators were 90.37%, 90.79% and 90.58% respectively, all above 90%, all achieving good results.

- When the number of instructions increases, the values of the evaluation indicators display a decline of 2% to 5%, which is in the range of acceptable.

- The average maximum response time is about 0.79s, which satisfies the real-time human-computer interaction.

- Balanced F values initially indicate good precision-recall performance, with performance deteriorating as the number of natural language instructions considered in the experiment increases.

4.3.3. Test experiments on complex natural language instruction parsing algorithms

In order to improve the scientific, applicability and extensibility of structured parsing algorithms for complex natural language instructions. In this paper, the resultant optimisation algorithm is added after the SDPA algorithm. This section continues to use control variables as well as random assignment, and 50, 100, 150...300 instructions are entered in turn for testing. The values of the evaluation metrics obtained and the maximum time to return results are shown in Table 8. Where Figure 9. clearly reflects the changes in the evaluation metrics, where P1, R1 and F1 represent the performance evaluation metrics before optimisation and P2, R2 and F2 represent the performance evaluation metrics after optimisation.

Table8. Complex natural language instruction test results

No.	Num	Precision	Recall	F-score	Time
1	50	89.14%	87.64%	88.59%	1.12s
		95.10%	**93.27%**	**94.18%**	**1.47s**
2	100	80.09%	81.36%	80.66%	1.46s
		89.81%	**90.65%**	**90.22%**	**1.48s**
3	150	78.02%	79.37%	78.68%	1.47s
		87.58%	**89.91%**	**88.73%**	**1.48s**
4	200	76.24%	77.36%	76.80%	1.52s
		85.43%	**86.67%**	**86.05%**	**1.48s**
5	250	74.21%	74.49%	74.35%	1.57s
		84.57%	**85.56%**	**85.06%**	**1.47s**
6	300	71.17%	71.29%	71.23%	1.70s
		82.29%	**84.41%**	**83.34%**	**1.49s**
Avg		78.15%	78.59%	78.37%	1.47s
		87.63%	**88.41%**	**87.93%**	**1.47s**

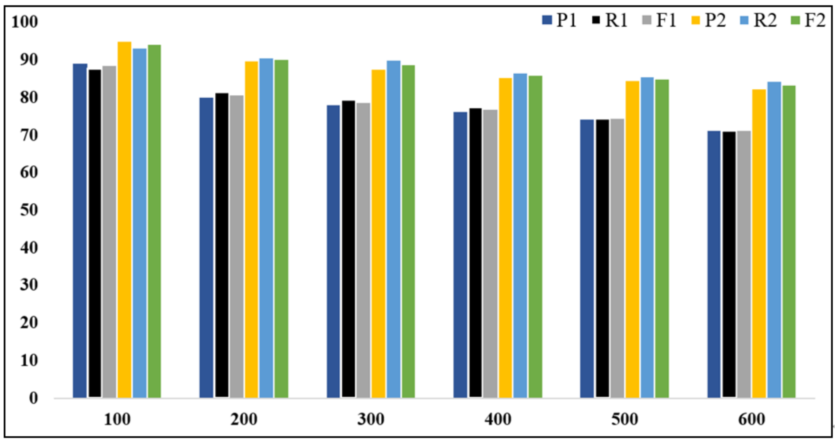

Figure 9. TICS performance evaluation bar chart

As can be seen from the figure: 443

- Either the SDPA algorithm alone on complex instruction test sets or the optimized SDPA algorithm on complex instruction test sets have better results, with the average P1 and R1 reaching 78.15% and 78.59%, and P2 and R2 reaching 87.63% and 88.41%.
- Algorithm performance all declined to some extent as the number of complex control instructions increased. The F-value decreases before optimisation were 7.93%, 1.98%, 1.88%, 2.45% and 3.12% respectively, with an average decrease of 3.47%. After optimisation, the F-values decreased by 3.96%, 1.49%, 2.68%, 0.99% and 1.72% respectively, with an average decrease of 2.17%. It is clear that the average decrease in F-value is significantly smaller than that of the pre-optimisation algorithm, indicating that the optimisation algorithm plays a significant role in the process of parsing complex natural language instructions, improving the scalability and scientific generalisation of the algorithm.
- The average response time is about 1.474s to meet the real-time human-computer interaction.

4.3.4. Text2SCADA combined service performance test experiment

In this section, 600 test commands were selected and entered into the Text2SCADA system in order of 100, 200, 300...600 using random assignment and control variables, the experimental results are shown in Table. 9, where Figure. 10 reflects the changes in the evaluation indicators.

Table9. Combined Algorithm Service Performance Testing

No.	Number	Precision	Recall	F-score	Time
1	100	98.26%	97.37%	97.81%	1.521s
2	200	94.62%	93.31%	93.96%	1.571s
3	300	91.23%	90.39%	90.81%	1.585s
4	400	85.23%	84.91%	85.07%	1.597s
5	500	83.58%	85.91%	84.73%	1.619s
6	600	82.68%	83.78%	83.23%	1.662s
Avg		**89.27%**	**989.28%**	**89.27%**	**1.593S**

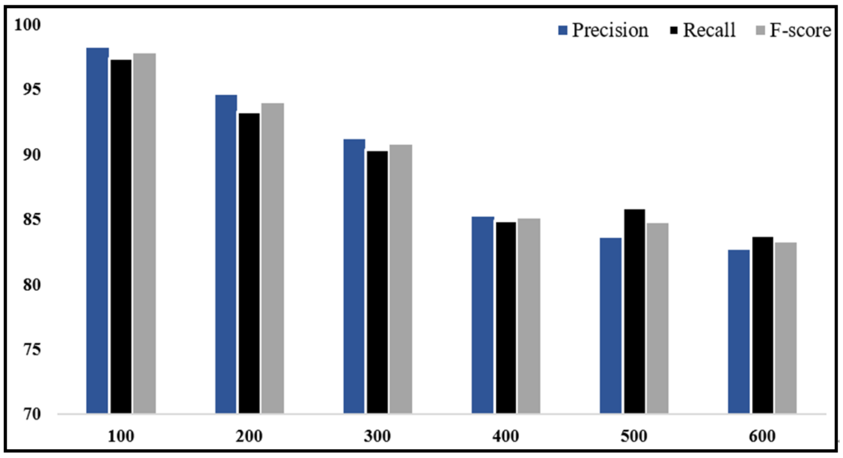

Figure 10. Combinatorial algorithm performance evaluation

From this the following insights can be obtained.

- The average Precision and Recall values for Text2SCADA under the command test set were 89.27% and 89.28% respectively. The overall accuracy is in the vicinity of 90%.

- The F-value gradually decreases, in the order of 3.91%, 3.15%, 5.74%, 0.34% and 1.5%, with an average decrease of 2.93%, mainly considering that as the number of test instructions increases, its testing effect will deteriorate to a certain extent, thus leading to a gradual decrease in F-value.

- The average response time of Text2SCADA is about 1.593s to meet the real-time human-computer interaction.

4.3.5. Comparison experiments

A total of 300 instructions were randomly selected and compared using the TF-IDF keyword extraction algorithm, Word2vec-based similarity algorithm, LSTM-based sentence similarity algorithm and template matching method and the algorithm of this paper. The cosine similarity threshold was 0.65. The successful recognition rate was used as the test index, the numerator was the number of successfully recognized instructions and the denominator was the total number of instructions, the results are shown in Table 10.

Table 10. Comparison experiments

Method	Success rate
TF-IDF	38.67%(116/300)
Word2vec	40.33%(121/300)
LSTM	43.67%(131/300)
Template Matching	63.67%(191/300)
Seq2Seq	70.33%(211/300)
Our Method(DGCNN)	96.67%(287/300)

TF-IDF keyword extraction algorithm can extract information from specified keywords, but it does not consider the case of multiple words with one meaning, and synonyms are used to construct links between them, so only 116 items can be recognized. Word2vec method mainly calculates the similarity between words, but cannot extract suitable keywords, and can only recognize some instructions in basic natural language instructions. LSTM takes sentences as units and calculates sentence meanings, which has a certain improvement in recognition, but the information extraction ability is average. The manual template matching method is based on manual rules and the extraction algorithm is written based on experience, which has a good success rate of recognition, but the method requires a lot of manpower to read the text and is not applicable to the actual human-computer interaction system, while the traditional Seq2Seq model has poor results in local perceptual ability and does not accomplish better information extraction. In this paper, we build a DGCNN-based information extraction model based on the TICS algorithm and SDPA algorithm, which makes the overall scalability and scientific generalization ability of Text2SCADA improved, and the success rate of command recognition is up to 96.67%.

4.3.6. Software and Example Testing

In this section, QT is used to write an agricultural human-machine interface as shown in Figure 11, relying on the Rapid SCADA cloud platform to implement the interaction algorithm. The parameter transfer is done using socket communication. We set the IP address as 103.45.156.173 and the port as 10086, and define three scenarios in the interface, namely, a greenhouse, a cornfield and a paddy field, where four field devices are set in the cornfield, namely, lights, air conditioners, cameras and sensors; lights, cameras, nozzles and sensors in the cornfield; and nozzles, cameras, lights and sensors in the paddy field. and sensors in the rice field. Each device will have a circular logo next to it, where gray represents the off state and green represents the on state. SCADA platform server, the corresponding result will be displayed on the interface.

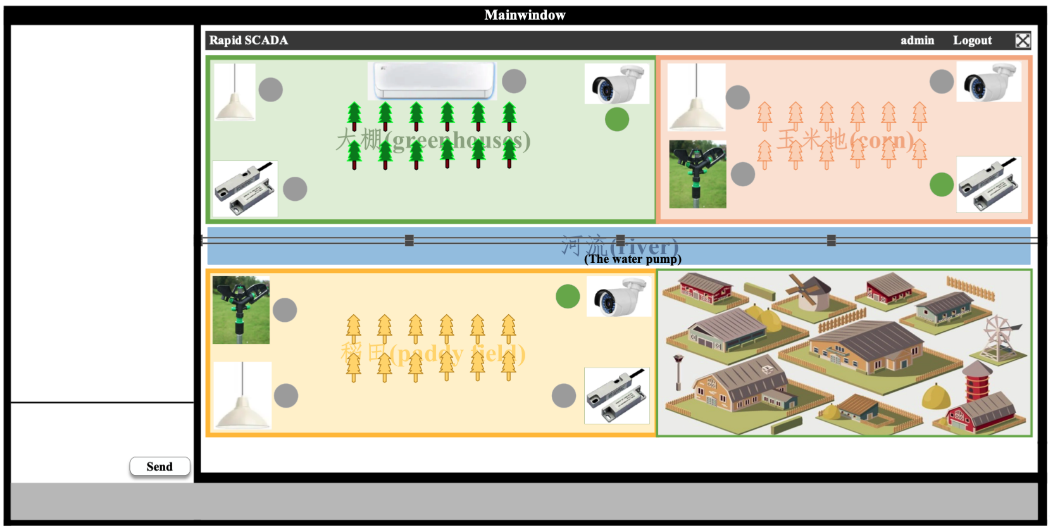

Figure 11. Human Machine Interaction Interface in Agriculture

Then, we tested four commands: "请帮我打开大棚和稻田的吊灯(Please help me turn on the chandelier of the shed and the rice field)", "请帮我关闭稻田的摄像头，然后打开玉米地的喷头，最后关闭大棚的摄像头(Please help me turn off the camera of the rice field, then turn on the nozzle of the cornfield, and finally turn off the camera of the shed)", "请帮我打开大棚的空调和传感器(Please Help me turn on the air conditioner and sensor of the shed)", "请帮我先打开到的摄像头，然后关闭稻田的吊灯，最后关闭玉米地的传感器和喷头(Please help me turn on the camera of to first, then turn off the chandelier of the rice field, and finally turn off the sensor and sprinkler of the cornfield)". The test results are shown in Figure 12.

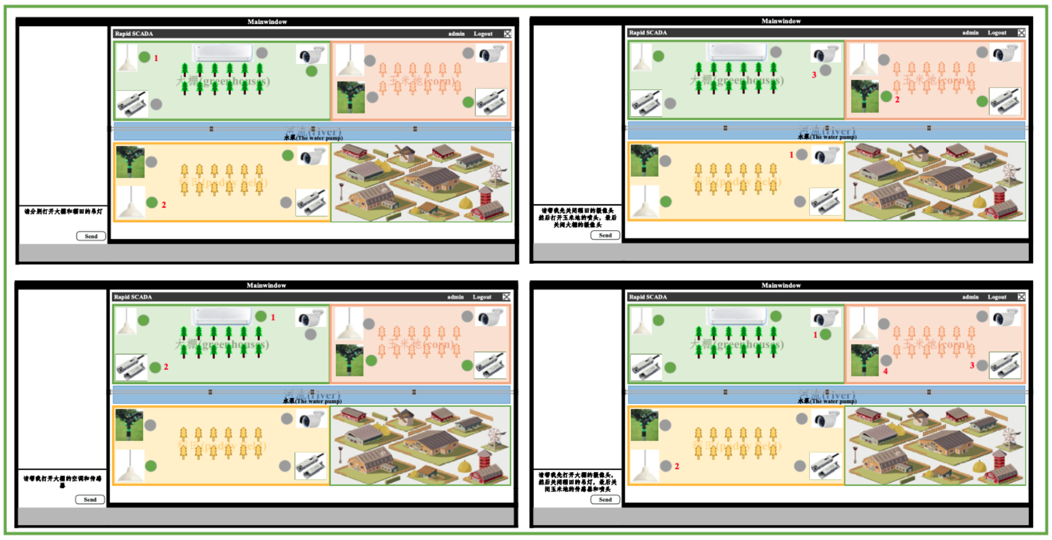

Figure 12. Presentation of natural language command test results on an agricultural HMI interface

Table11. Example diagram of natural language instruction parsing

No.	Field	Original Instructions	Intermediate Language
1	Agricultural	先打开温室的加热机，然后关闭大棚的喷雾器。(Turn on the greenhouse heater first, then turn off the sprayer in the greenhouse)	[{'action': '打开', 'location': '温室', 'object': '加热机'}, {'action': '关闭', 'location': '大棚', 'object': '喷雾器'}]
2	Industrial	先打开高温炉的电机，然后关闭热电厂的阀门。(Turn on the motor of the high temperature furnace first, then close the valve of the thermal power plant.)	[{'action': '打开', 'location': '高温炉', 'object': '电机'}, {'action': '关闭', 'location': '热电厂', 'object': '阀门'}]
3	Smart home	请帮我先打开客厅的电风扇，然后再帮我打开一下卧室的空调，谢谢你了。(Please help me turn on the electric fan in the living room first, and then turn on the air conditioner in the bedroom, thank you very much.)	[{'action': '打开', 'location': '客厅', 'object': '电风扇'}, {'action': '打开', 'location': '卧室', 'object': '空调'}]

Conclusion

The main contribution of this research is to propose a natural language interface (Text-to-SCADA) for distributed SCADA systems, which first distinguishes basic natural language instructions from complex natural language instructions using lexical annotation, then parses basic natural language instructions by the TICS algorithm and complex natural language instructions by the SDPA algorithm, and finally understands the natural language input from the user instructions. The CNLQTS dataset is provided through crowdsourcing to test the performance of the interface, and finally a DGCNN-based information extraction model is constructed to enhance the scientific generalization ability of the model using deep learning methods. After that, more complex natural language instructions will be investigated and intermediate language formats will be enriched. Furthermore, future work will consider encoding instructions in the form of graph neural networks to build knowledge graphs to improve the accuracy of complex queries and support higher scalability.

Acknowledgements This work was supported by Key RD Project of Nanning Science and Technology Bureau, Guangxi, China, "Research and Development of Face Recognition

Platform Based on GPU Fragmentation Virtualization Heterogeneous Resource Scheduling (20203117)"

References

1. Daneels, Axel, and Wayne Salter. "What is SCADA?." (1999).
2. Cannan, James, and Huosheng Hu. "Human-machine interaction (HMI): A survey." University of Essex (2011): 27.
3. Preece, Jenny, Yvonne Rogers, Helen Sharp, David Benyon, Simon Holland, and Tom Carey. Human-computer interaction. Addison-Wesley Longman Ltd., 1994.
4. Allen, James. Natural language understanding. Benjamin-Cummings Publishing Co., Inc., 1995.
5. Li, Fei, and Hosagrahar V. Jagadish. "Constructing an interactive natural language interface for relational databases." Proceedings of the VLDB Endowment 8, no. 1 (2014): 73-84.
6. Androutsopoulos I，Ritchie G D，Thanisch P．Natural language interfaces to databases An introduction EJ]．Natural Language Engineering，1995，1(1)：29—81.
7. oods W A．Progress in natural language understanding：An application to lunar geology[c]Proc of the National Computer Conf and Exposition．New York：ACM，1973：44]一45.
8. Sacerdoti E D．Language access to distributed data with error recovery Proc of the 5th Int Joint Conf on Artificial Intelligence．San Francisco，CA：Morgan Kaufmann，1977：196-202.
9. Warren D H D，Pereira F C N．An efficient easily adaptable system for interpreting natural language queries ㅁ]．American Journal of Computational Linguistics，1982，8(3 4)：
10. Thompson B H，Thompson F B．Introducing ask，a simple knowledgeable system[c] Proc of the 1 st Conf on Applied Natural Language Processing．Stroudsburg，PA：ACL，1983.
11. Affolter K，Stockinger K，Bernstein A．A comparative survey of recent natural language interfaces for databases[J]．The VLDB Journal，2019，28(5)：793—819 .
12. Blunschi L，Jossen C，Kossmann D，et a1．SODA：Generating SQL for business users[J]．Proceedings of the VLDB Endowment，2012，5(10)：932-943.
13. Zenz G，Zhou Xuan，Minack E，et a1．From keywords to semantic queries incremental query construction on the semanticweb[J]．Journal of Web Semantics，2009.
14. Damljanovic D，Tablan V，Bontcheva K．A text—based query interface to OWL ontologies It] Proe of the 6th Language Resources and Evaluation Conf(LREC)．Paris：European Language Resources Association(ELRA)，2008：205—212.
15. Zheng Weiguo，Cheng Hong，Zou Lei．et a1．Natural language question answering：Let users talk with the knowledge graph[c]Proc of the 2017 ACM on Conf on Information and Knowledge Management．New York.
16. Saha D，Floratou A，Sankaranarayanan K．et a1．ATHENA：An ontology-driven system for natural language querying over relational data stores Eft]．Proceedings of the VLDB Endowment，2016，9(12)：1209-122.
17. Li Fei，Jagadish H．Constructing an interactive natural language interface for relational databases[J]．Proceedings of the VLDB Endowment，2014，8(1)：73—84.
18. Bernstein A，Kaufmann E，Kaiser C．Querying the semantic web with Ginseng：A guided input natural language search engine[COL Proc of the 15th Workshop on Information Technology and Systems．2005：112—126[2020—03—12]．https：files．ifi．uzh．ch ddis oldweb dis staff goehring btw files BernsteinEtAl-Ginseng—WITS2005.
19. Song Dezhao，Schilder F，Smiley C，et a1．TR Discover：A natural language interface for querying and analyzing interlinked datasets[c] Proc of the Int Semantic Web Conf (ISWC)．Berlin：Springer，2015：21—3.
20. Brad F，Iacob R C A，HOSU I A，et a1．Dataset for a neural natural language interface for databases(NNLIDB)[COL] Proc of the 8th Int Joint Conf on Natural Language Processing．Asian Federation of Natural Language Processing，2017：906—914[2020—03—12]．https：www．anthology 117—109I．pdf.
21. Lecun Y，Bengio Y，Hinton G．Deep learning[J]．Nature，2015，521(7553)：436—444.
22. Yin Pengcheng，Neubig G．TRANX：A transition—based neural abstract syntax parser for semantic parsing and code generation[c] Proc of the 2018 Conf on Empirical Methods in Natural Language Processing：System Demonstrations．Stroudsburg，PA：ACL，2018 .